UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

PROYECTO FIN DE GRADO

APLICACIÓN DE LAS CÁMARAS 3D AL RECONOCIMIENTO DE ACTIVIDADES

AUTOR: ERIC MAGRO VIFORCOS

TUTOR: DR. ARTURO DE LA ESCALERA HUESO

SEPTIEMBRE 2012

"Los imposibles de hoy,

serán posibles mañana."

Konstantin Eduardovich Tsiolkovsky, 1935.

Agradecimientos

Quería agradecer a Arturo de la Escalera su apoyo y dedicación durante la realización del proyecto, por poder contar con el cuando no encontraba la manera de seguir y ofrecerme el material y un lugar para realizar el proyecto.

Agradecer a mis compañeros, en especial a Santiago Morante, Borja Martin y Roberto Montero por aquellas sesiones en las que discutíamos posibles soluciones a los problemas.

A mi novia por soportar aquellas conversaciones sobre el proyecto, en las que no se enteraba de nada pero me servían para desahogarme y aclarar las ideas.

Y por ultimo agradecer a mis padres y mi hermano el apoyo recibido durante toda la carrera y no dudar a prestarse a realizar las pruebas de esqueletización en el momento que se lo pidiera.

INDICE GENERAL

ABSTRACT 11

ABREVIATURAS 13

1. INTRODUCCIÓN 15

1.1 MOTIVACIÓN Y CONTEXTO DEL PROYECTO 15

1.2 OBJETIVOS 16

1.3 INTRODUCCIÓN A LA VISIÓN POR COMPUTADOR 16

1.4 OBTENCIÓN DE IMÁGENES DE DISPARIDAD MEDIANTE CÁMARA ESTÉREO 19

2. DESCRIPCIÓN GENERAL DE KINECT 21

2.1. ESPECIFICACIONES TÉCNICAS 21

2.2. MÉTODO DE OBTENCIÓN DE LA PROFUNDIDAD 25

3. HERRAMIENTAS UTILIZADAS 29

3.1. OPENCV 29

3.2. VISUAL STUDIO 2010 30

4. DEFINICIÓN DE DRIVERS 33

4.1. KINECT FOR WINDOWS SDK 33

4.2. OPENNI 44

5. ESQUELETIZACIÓN MEDIANTE CÁMARA ESTÉREO. 57

6. COMPARACIÓN ENTRE SDK Y OPENNI 59

7. USOS Y APLICACIONES DE KINECT 67

8. CONCLUSIONES 75

9. POSIBLES LÍNEAS DE FUTUROS TRABAJOS 77

PRESUPUESTO 79

BIBLIOGRAFÍA 81

Resumen

El objetivo perseguido al realizar este proyecto es dar a conocer a todos los usuarios que se vayan a iniciar al uso de Kinect, en especial a la adquisición de imágenes, y al uso del reconocimiento y esqueletización de las personas, de las características, ventajas e inconvenientes de dos de las bibliotecas para el manejo de la cámara.

Además se estudiarán las funciones de OpenNI para conseguir la detección de personas mediante la adquisición de imágenes estereoscópicas.

Se mostrará en qué lugares se está utilizando la cámara para demostrar el potencial de Kinect fuera del ámbito para el que se diseñó, los videojuegos. Fomentar el uso de herramientas que con un precio reducido pueden aportar mucha información y lograr que cualquier persona pueda investigar y desarrollar aplicaciones desde su propia casa.

Palabras clave:

visión por computador, OpenCV, Kinect, Kinect for Windows SDK, OpenNI, estéreo, esqueletización, detección de personas.

Abstract

The objective of this project is going to be initiate everybody that will be to initiate the Kinect's use, characteristics, advantages and disadvantages of two libraries to handle the camera, especially image acquisition, recognition and tracking people.

Also we will try to use OpenNI functions to achieve the people detection through the acquisition of stereoscopic images.

We'll show where the camera is being used to demonstrate the potential of Kinect outside the scope which it was designed, video games. Promote the use of tools that can provide a lot of information and that anyone can research and develop applications from your own home at a reduced price.

Keywords:

computer vision, OpenCV, Kinect, Kinect for Windows SDK, OpenNI, stereo, tracking, detection of persons.

Abreviaturas

SDK	**S**oftware **D**evelopment **K**it
OpenNI	**Open N**atural **I**nterface
CCD	**C**harge **C**ouple **D**evice
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
RGB	**R**ed **G**reen **B**lue
OpenCV	**Open** Source **C**omputer **V**ision
NIR	**N**ear **I**nfra**r**ed
FPS	**F**otos **P**or **S**egundo
USB	**U**niversal **S**erial **B**us
SO	**S**istema **O**perativo
BSD	**B**erkeley **S**oftware **D**istribution
API	**A**pplication **P**rogramming **I**nterface
MFC	**M**icrosoft **F**oundation **C**lass

1. Introducción

En este capítulo se mostrarán de los motivos de la elección del proyecto a través de la evolución y la tecnología emergente en la actualidad. También se detallará los objetivos del proyecto y se realizará una introducción a la visión por computador.

1.1 Motivación y contexto del proyecto

En la actualidad existe una gran expansión de la visión por computador y sus aplicaciones tanto en la vida cotidiana como en el ámbito de la investigación.

Con la reciente aparición de las "Smart TV" el uso del mando a distancia comienza a desaparecer debido a la incorporación de cámaras capaces de detectar el movimiento del usuario para su manejo.

Pero existen más aplicaciones en las que se pueden utilizar el cuerpo como mecanismo de control. Esta tecnología también se encuentra aplicada en los videojuegos como es el caso de la cámara Kinect de Microsoft o su antecesora Eye Toy de Sony.

Pero la detección del movimiento del usuario no solo tiene aplicaciones interactivas sino que actualmente también se utiliza para el manejo de robot mediante gestos, aplicaciones en biomedicina para analizar el movimiento de las personas…

Debido al auge del uso del cuerpo humano para el manejo de sistemas crea la necesidad de la investigación de su funcionamiento, centrándose en la cámara Kinect y sus librerías para ordenador.

Todo lo referente a las aplicaciones de Kinect y su análisis del esqueleto se desarrollará más adelante en el apartado de "Usos y aplicaciones de Kinect".

1.2 Objetivos

El objetivo principal de proyecto será el análisis y comparación de las librerías "Kinect for Windows SDK" y "OpenNI".

Para ello se crearán aplicaciones en las que se utilizarán las funciones de captura de imágenes y esqueletización y las se compararán entre ellas para ver las diferencias.

Serán analizadas las funciones de ambas librerías para ver las diferencias entre lo que permite las librerías oficiales de Microsoft (Kinect for Windows SDK) y las no oficiales (OpenNI).

Además se intentarán incorporar las imágenes estereoscópicas a OpenNI y realizar la esqueletización utilizando esas imágenes.

1.3 Introducción a la visión por computador

La visión por computador, también conocida como visión artificial, consiste en que partiendo del mundo real podamos extraer la información necesaria a partir de las imágenes y las procesemos en un computador. Este sistema intenta imitar el procedimiento y realizar algunas de las tareas de la visión humana, desde la simple detección de objetos sencillos a la interpretación y análisis de entornos tridimensionales.

En un primer momento toda la información visual consiste en energía luminosa procedente de los objetos del entorno. En el caso de la visión humana es nuestro ojo el encargado de recibir esa energía luminosa y transformarlas en señales electroquímica que envía al cerebro. Dentro del ojo, la parte que se encarga de la transformación es la retina, que posee dos tipos de receptores (conos y bastones) que se encarga de la percepción del color y de las intensidades luminosas.

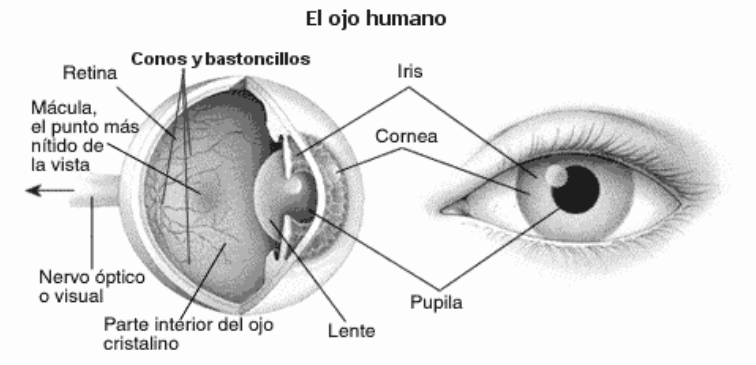

Ilustración 1. Partes del ojo humano.

En la visión artificial la encargada de realizar la tarea realizada por el ojo es una cámara de vídeo (o cualquier otro dispositivo similar), que se encarga de convertir la energía luminosa en corriente eléctrica (cámaras analógicas) o señales digitales (cámaras digitales). Si la cámara es analógica deberá pasar por un proceso de muestreo y digitalización para poder ser procesada por un computador.

Para poder capturar las imágenes las cámaras poseen matrices de receptores. Cada elemento de la matriz se le conoce como píxel y el valor que toma al incidir sobre el la luz se llama "intensidad" o "nivel de gris". Los elementos fotosensivos pueden ser de diversas tecnologías como CCD o CMOS. Dependiendo si las imágenes que deseamos conseguir son a color o escala de gris, deberemos usar 3 elementos fotosensivos por pixel si deseamos que sea a color.

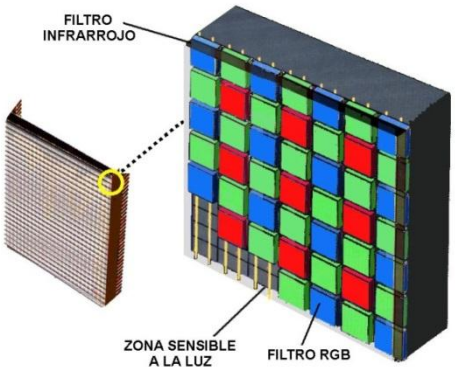

Ilustración 2. Sensor CCD formato RGB

La visión del ser humano (y amínales) no muestrea de forma uniformemente dentro de su campo visual, sino que lo realiza de forma más densa en la fóvea y menos en la periferia. Al utilizar cámaras este comportamiento no se suele imitar aunque si se deseara se podría realizar, denominándose visión foveal, donde algunos objetos se verán nítidamente mientras que si nos alejamos de éste, la imagen se irá distorsionando.

Las razones para que este comportamiento no se imite son las siguientes:

- Desconocimiento del mecanismo humano para centrarse en puntos de atención
- Mientras que la visión humana no tiene los fotoreceptores uniformes, perciben con una resolución única.

Existen varios campos que se encontrarían englobados dentro de la visión por computador, que dependiendo del problema concreto tendría más de uno que de otros. Los campos son los siguientes:

- Inteligencia artificial: que consiste en la interpretación del entorno, el aprendizaje y razonamiento cognitivo de la máquina por si solo.
- Procesamiento de imágenes: mediante procesos de transformación de las imágenes obtenemos otras de mayor calidad o destacando los detalles de importancia.
- Reconocimiento de patrones: para la clasificación de objetos según patrones seleccionados.
- Gráficos por computador: consiste en que partiendo de mundo tridimensional, plasmarlo en un mundo bidimensional.

La visión artificial está cobrando gran interés entre el campo científico-técnico debido a su gran número de aplicaciones y reducción en costes tanto de mano de obra como debido a la devolución de los productos en el campo de producción. Además, su uso está extendido tanto al análisis de imágenes médicas, a la robótica, navegación autónoma…

1.4 Obtención de imágenes de disparidad mediante cámara estéreo

La visión estereoscópica está basada en la visión humana. Al igual que los humanos, posee dos cámaras que toman imágenes en 2 dimensiones y tras un proceso de análisis se transforman esas imágenes en otra que contiene los datos de profundidad. Para ello es necesario que ambas cámaras permanezcan a una distancia fija y tomen las imágenes al mismo tiempo.

Ilustración 3. Cámara estereoscópica

Para obtener la imagen estéreo se toman imágenes desde las dos cámaras y se buscan la correspondencia entre los puntos de las dos cámaras. A partir de la diferencia de posición de esos puntos en ambas imágenes se puede sacar la profundidad mediante triangulación.

El fundamento para calcular la distancia de un punto respecto de la cámara consiste en que los puntos más cercanos aparecerán en posiciones más dispares en una imagen respecto a la otra. Sin embargo, aquellos puntos más alejados aparecerán en posiciones más cercanas llegando a ser la misma para puntos situados en el infinito.

La distancia entre ambas lentes en la cámara influye directamente en la disparidad en las imágenes de forma proporcional, si aumenta la distancia entre las lentes aumenta la disparidad, y por tanto la distancia que podremos medir.

Para poder calcular la disparidad habrá que conocer qué píxeles en una imagen corresponden a los de la otra, la forma de encontrar estos puntos de interés influirá en los resultados obtenidos.

Existen dos formas de obtención de mapas de disparidad:

Emparejamiento discreto: se obtiene la imagen buscando puntos de interés y generando la imagen de disparidad usando únicamente estos puntos

Emparejamiento denso: la imagen de disparidad se obtiene analizando toda la imagen, por lo que el método es mucho más lento que el anterior.

En ambos casos, antes de buscar la correspondencia entre ambas imágenes hay que pasar por un proceso de calibración de la cámara. En la librería OpenCV existen funciones para calcular los parámetros de calibración de la cámara mediante el uso de un tablero de ajedrez. Para la calibración se utiliza un tablero debido a su sencilla geometría. Mediante la calibración de la cámara conseguimos eliminar las aberraciones.

Una vez calibrada la cámara y haber obtenido la imagen de disparidad (véase ilustración 4) se procede a obtener la distancia de los puntos mediante triangulación.

Ilustración 4. Imagen estéreo

2. Descripción general de Kinect

Kinect, cuyo nombre originalmente fue "Proyecto Natal", es un dispositivo creado por Alex Kipman y desarrollado por Microsoft para su videoconsola Xbox 360.

Este dispositivo fue anunciado por primera vez en junio de 2009 en la Electronic Entertainment Expo 2009 y anunciado como la nueva generación de entretenimiento en el hogar. Esto es debido a que mediante el uso de la cámara puedes controlar mediante movimientos de tu cuerpo las acciones del jugador y los menús de juego.

2.1. Especificaciones técnicas

El dispositivo Kinect (véase figura 5) dispone de dos cámaras, RGB y NIR, y una fuente de luz infrarroja que permite que la cámara NIR obtenga sus datos incluso en ausencia de luz.

Mientras que la cámara RGB se encarga de obtener la información de color de todo aquello situado en su campo de visión, la cámara infrarroja de corto alcance se encarga de obtener la información de profundidad. Las dos lentes exteriores corresponden con la información infrarroja mientras que la lente situada en medio corresponde con la cámara a color.

Ilustración 5. Dispositivo Kinect

Además de las dos cámaras y el emisor, el dispositivo cuenta con un motor que permite el movimiento ascendente y descendente y cuatro micrófonos situados horizontalmente para ubicar la fuente de sonido y suprimir el ruido ambiente. En la imagen siguiente (Figura 6) se observan los componentes de la cámara.

Ilustración 6. Despiece Kinect

Las especificaciones técnicas que posee la cámara son las siguientes:

- **Sensores y actuadores**

  Cuatro micrófonos para localizar el foco emisor y anular el ruido ambiente. Situados en la parte horizontal inferior.

  Dos cámaras de tecnología CMOS (tanto color como infrarrojos) y proyector de infrarrojos (lente derecha).

  Ventilador para la disipación del calor interno.

  Motor para mover la cámara en sentido vertical.

  Acelerador de tres ejes para aumentar la precisión en el movimiento del motor.

  Chip PrimeSense PS1080-A2 para procesamiento de los mapas.

- **Rango de visión**

Campo de visión horizontal de 57 grados.

Campo de visión vertical de 43 grados.

Rango de inclinación debido a los motores de ±27 grados.

Profundidad de detección desde 0.8 m hasta 8 m.

- **Flujo de datos**

Sensor de profundidad: 320 x 240 y 16 bit por pixel, velocidad 30 fps

Sensor a color: 640 x 480 y 32 bit por pixel, velocidad 30 fps

Audio: 16 bits y velocidad 16 kHz

Tras su éxito en ventas y la aparición de drivers no oficiales para su uso en el ordenador, Microsoft lanzo sus propios drivers a los que llamó "Kinect for Windows SDK". Tras su lanzamiento de la versión beta y 1.0 han lazado su versión 1.5.

Debido a su gran expansión y uso para computadores, decidieron sacar un nuevo modelo de Kinect en el que la alimentación se hacía directamente desde el USB olvidándonos de la conexión al enchufe. Este nuevo modelo, diseñado específicamente para ordenador y no compatible con Xbox, posee características diferentes a su antecesora.

- **Sensores y actuadores**

Permanecen igual que su antecesora.

- **Campo de visión**

Solo se modifica el rango de profundidad apareciendo dos modos de funcionamiento:

En el modo Default Mode recogemos información desde los 0.8 m hasta 4 m de forma óptima, y desde los 4m hasta los 8 m recogemos información no optimizada.

En el modo “Near Mode” medimos desde los 0.4m hasta 3 metros (mejor calidad a los 2m) y desde los 3m hasta 8m de forma no optima.

En la imagen siguiente (Figura 7) podemos comparar las distancias de ambos modos.

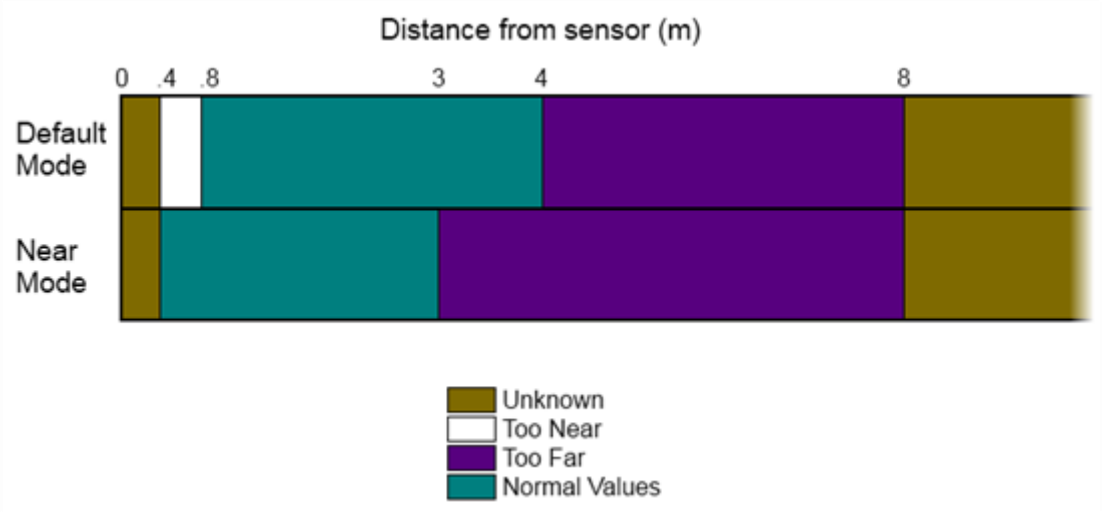

Ilustración 7. Modos de funcionamiento.

- **Flujo de datos**

Sensor de profundidad: 80x60, 320x240, 640x480 y velocidad de 30 fps

Sensor a color: 640x480 a 32 bits de color y 30 fps, 1280x960 RGB y 12 fps, 640x480 Raw YUV a 15 fps.

Sensor audio: 32 y 64 bits en función del SO.

2.2. Método de obtención de la profundidad

El secreto del bajo precio de la cámara Kinect es su mecanismo para la obtención de la profundidad. La mayor parte de las cámaras infrarrojas, para la obtención de la profundidad, se basan en "tiempo de vuelo". Estas cámaras emiten un pulso infrarrojo que al rebotar en un objeto vuelve a la cámara, la distancia del objeto a la cámara se calcula midiendo el tiempo desde que se lanzó el pulso hasta que se recibió. El precio elevado de estas cámaras reside en la dificultad de medir el intervalo del pulso luminoso (debido a que viaja a la velocidad de la luz), por lo que los cronómetros incorporados en las cámaras tienen que medir tiempos reducidos.

Para medir la distancia en aquellas cámaras que emiten luz infrarroja constantemente se utiliza el desfase producido en la longitud de onda entre el rayo emitido y el rayo recibido.

A diferencia de los dos métodos anteriores, la tecnología utilizada por Kinect no se centra solo en la posición del objeto, sino también en la detección y codificación de los haces de luz reflejada por los objetos.

Para obtener los datos de profundidad, la cámara emite un patrón infrarrojo conocido que captaremos desde el sensor infrarrojo. Para calcular la distancia de cada punto mediremos la deformación de los puntos que componen esa malla.

Ilustración 8. Malla de puntos infrarrojos.

Mediante el chip de procesamiento de las imágenes (PrimeSense PS1080-A2) se descompone la imagen y la codificación de la constelación infrarroja, eliminando el ruido producida por los rayos infrarrojos reflejados por la luz solar.

El cálculo de la distancia al objeto se realiza mediante triangulación de cada punto de la malla infrarroja de partida y su correspondiente punto en la malla recibida. Mediante el cálculo de todos los puntos del patrón se genera el mapa de profundidad.

El patrón de luz emitida por la cámara se puede observar en las imágenes cuando no se muestran a máxima resolución.

Ilustración 9. Patrón infrarrojo emitido.

El tamaño de los puntos de la constelación infrarroja nos proporciona información de la distancia al sensor. Se pueden distinguir varias zonas de funcionamiento, en las que la precisión del sensor variará según la zona.

- Desde los 0.8 m hasta los 1.2 m la precisión de las medidas es muy alta debido a la gran cantidad de puntos que contendrán los objetos, por lo que la precisión a la hora de conseguir la forma de los objetos será alta.
- Desde 1.2 m hasta los 2 m las medidas tomadas poseen algunos puntos menos pero la precisión en las medidas siguen siendo buenas.
- A partir de los 2m la perdida de información comienza a ser significativa, pudiendo detectar las siluetas pero no algunas formas de los objetos. Cuando llegamos a los 8m la perdida de información es lo suficientemente grande como para desestimar la detección de personas.

3. Herramientas utilizadas

En el capítulo 3 detallaremos las herramientas utilizadas en la elaboración del proyecto. Estas herramientas fueron elegidas debido a que habían sido utilizadas durante el trascurso de la carrera, por lo que se ahorraba tiempo al no tener que aprender su uso.

3.1. OpenCV

OpenCV es una librería de funciones de visión artificial para el tratamiento de imágenes. Fue desarrollada en 1999 por la compañía Intel Corporation y publicada bajo licencia BSD, lo que permite que sea usado con fines comerciales e investigación.

Las funciones y clases que componen la librería están escritas en C y C++ y además son multiplataforma, pudiéndose utilizar tanto en aplicaciones para GNU/Linux, MacOS y Microsoft.

Una de las principales ventajas de ser una librería de código abierto es su gran uso por parte de comunidades que comparten sus experiencias y códigos con los que ayudar en su uso y aprendizaje

La principal característica de OpenCV es la funcionalidad, claridad y sencillez de uso de sus funciones. Los algoritmos están basados en estructuras de datos. Dispone de más de 500 funciones dedicadas a:

- Procesamiento general de imágenes.
- Segmentación
- Descriptores de geometría
- Imágenes Piramidales
- Calibración de cámaras
- Transformaciones geométricas
- Detección de rostros
- Maquinas de aprendizajes
- Etc

OpenCV puede descargarse desde su página oficial (opencv.willowgarage.com). En nuestro caso utilizaremos la versión 2.2 y la utilizaremos para mostrar las imágenes obtenidas mediante Kinect y representar los puntos del tracking de cada persona.

3.2. Visual Studio 2010

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE), diseñado para el sistema operativo de Windows. En Visual Studio podemos programar tanto en C++, C#, J#, NET, Basic…

Existe una versión de Visual Studio que Microsoft proporciona de manera gratuita desde su página oficial. Estos productos los han denominado Express y son ediciones básicas en las que podemos encontrar cada lenguaje de programación por separado. En la versión de 2010 podemos encontrar:

- Visual Web Developer 2010 Express.
- Visual Studio 2010 Express para Windows Phone.
- Visual Basic 2010 Express.
- Visual C# 2010 Express.
- Visual C++ 2010 Express

Entre las novedades que ofrece la versión 2010 esta la incorporación de herramientas para el desarrollo de aplicaciones para Windows 7 y la capacidad de utilizar múltiples monitores. Pero Windows 7 no es la única plataforma, también podemos realizar aplicaciones para Windows Phone 7 entre otros.

Una vez tengamos instaladas las librerías de OpenCV y la librería de Kinect que vayamos a utilizar en ese momento, procederemos a agregarlas al proyecto de Visual Studio. Para la realización del proyecto utilizaremos “aplicación de consola de Win32” en el apartado de Visual C++. A la hora de agregar las librerías lo haremos de la forma siguiente:

Para agregar las librerías de OpenCV:

- Seleccionaremos el nombre de nuestro proyecto y abriremos el menú de propiedades.

- En el apartado de directorios de archivos de inclusión (en directorios de VC++) añadiremos:
  C:\OpenCV2.2\include
  C:\OpenCV2.2\include\opencv
  C:\OpenCV2.2\include\opencv2

- En el apartado de directorios de archivos de bibliotecas añadiremos:
  C:\OpenCV2.2\lib
- A continuación iremos a Vinculador/Entrada y añadiremos en dependencias adicionales la siguiente lista:
  opencv_calib3d220d.lib
  opencv_contrib220d.lib
  opencv_core220d.lib
  opencv_features2d220d.lib
  opencv_ffmpeg220d.lib
  opencv_flann220d.lib
  opencv_gpu220d.lib
  opencv_highgui220d.lib
  opencv_imgproc220d.lib
  opencv_legacy220d.lib
  opencv_ml220d.lib
  opencv_objdetect220d.lib
  opencv_video220d.lib
- Para finalizar añadiremos los “includes” en el código:
  #include <cv.h>
  #include <highgui.h>
  #include <ml.h>

Para agregar las librerías de Kinect for Windows SDK:

- Seleccionaremos el nombre de nuestro proyecto y abriremos el menú de propiedades.
- En el apartado de directorios de archivos de inclusión (en directorios de VC++) añadiremos:
  C:\Program Files\Microsoft SDKs\Kinect\v1.0\inc
- En el apartado de directorios de archivos de bibliotecas añadiremos:
  C:\Program Files\Microsoft SDKs\Kinect\v1.0\lib\x86
- A continuación iremos a Vinculador/Entrada y añadiremos en dependencias adicionales:
  Kinect10.lib
- Para finalizar añadiremos los "includes" en el código:
  #include <NuiApi.h>

Para agregar las librerías de OpenNI:

- En el apartado de propiedades del proyecto abriremos C++/General y añadiremos lo siguiente en la parte de directorios de inclusión adicionales:
  C:\Program Files\OpenNI\Include
- En Vinculador/General añadiremos en directorios de bibliotecas adicionales:
  C:\Program Files\OpenNI\Lib
- A continuación iremos a Vinculador/Entrada y añadiremos en dependencias adicionales:
  OpenNI.lib
- Para finalizar añadiremos los "includes" en el código:
  #include "XnCppWrapper.h" (por usar las funciones de C++)

4. Definición de drivers

Existen diferentes drivers para para el uso de Kinect mediante un ordenador. Entre ellas se encuentra Kinect for Windows SDK, OpenNI, OpenKinect o Libfreenect. Para el desarrollo del proyecto se eligió el uso de Kinect for Windows SDK y OpenNI, debido a la gran similitud entre ambos programas y su capacidad en la detección de personas.

4.1. Kinect for Windows SDK

La primera versión oficial del driver de Kinect apareció de forma gratuita en Junio de 2011. Esta versión (SDK beta) estaba diseñada para el desarrollo de aplicaciones para Windows 7. Los lenguajes de programación posibles son C++, C# y Visual Basic. Además, incorporaba una serie de ejemplos para ayudar al usuario en la tarea de comprensión de las funciones.

Tras el lanzamiento de su segunda versión beta, en Febrero de 2012 sacaron su primera versión 1.0. Esta nueva versión nos proporciona la posibilidad de hacer comerciales aquellas aplicaciones que desarrollemos para Windows. Los cambios sufridos respecto a las versiones anteriores son:

- Conexión de hasta 4 sensores Kinect al mismo tiempo, siempre que el equipo sea lo suficientemente potente y se puedan conectar cada una a un puerto sin usar concentradores de USB.
- Aparición del modo Near, disponible únicamente para la versión de Kinect para PC, por lo que en su antecesora (Kinect Xbox 360) no funcionará.
- Aumento en la robustez en la estabilidad del conductor, tiempos de ejecución y recepción de audio.
- Modificación en el nombre de algunas funciones, cabeceras y archivos DLL.
- Cambio en la resolución de la imagen RGB de 1280x1024 a 1280x960
- Calculo de forma correcta de las imágenes tomadas por segundo.
- Mejoras y aumento de funciones para cálculo y conversión de mapas.

- Mejoras en el reconocimiento de audio.
- Actualización del modelo de audio para obtener valores de confianza mayor.
- Cancelación del eco.
- Cambios en la inicialización del audio.
- Etc.

La versión que utilizaremos durante el proyecto será el SDK v.1.0, sin embargo existe la versión 1.5 que fue actualizada el día 21 de Mayo de 2012. Esta nueva versión incorpora funciones para el reconocimiento facial.

4.1.1. Arquitectura.

Kinect for Windows SDK (a partir de ahora SDK) nos proporciona acceso tanto a las imágenes de color, profundidad y los datos de audio. La interacción entre el hardware y software con la aplicación se muestra en la siguiente figura.

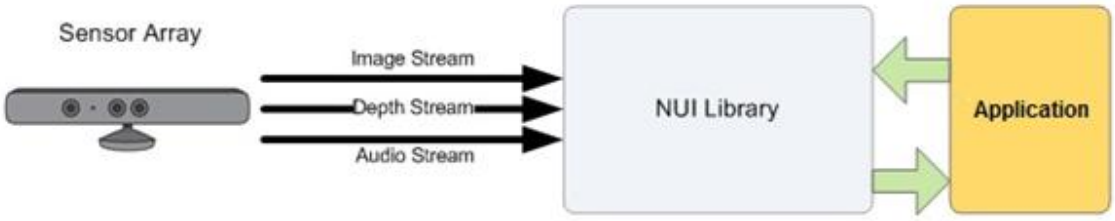

Ilustración 10. Interacción Kinect con las aplicaciones.

El SDK está compuesto de varios niveles (véase ilustración 11).

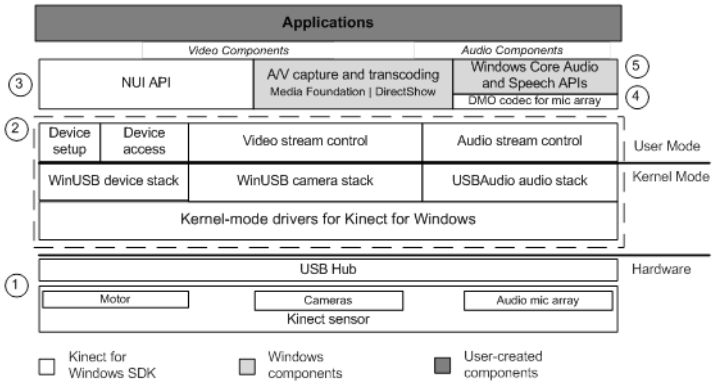

Ilustración 11. Arquitectura SDK

En el nivel de hardware tenemos los sensores de la Kinect, estos son el motor, la cámara RGB, el sensor y receptor infrarrojo y los micrófonos, además del concentrador de USB, que permite la conexión de sensores con el ordenador.

En el nivel de drivers tenemos lo relacionado a la adquisición de los datos de imagen y profundidad, el acceso a los datos de audio y la habilitación de conexión de más de una Kinect al mismo tiempo.

En siguiente nivel tenemos el interface entre la aplicación y las funciones que nos proporciona acceso a los datos.

4.1.2. Tipos de imágenes.

Para poder mostrar las imágenes, debemos haber inicializado antes que tipo de datos vamos a querer. Las opciones que nos permite el SDK son las siguientes:

- Color: proporciona las imágenes a color capturadas por la lente.
- Depth: proporciona las imágenes con los datos de profundidad.
- Depth and player index: contiene las imágenes de profundidad con la silueta de las personas detectadas superpuestas a la profundidad
- Skeleton: proporciona la posición de los esqueletos detectados y sus articulaciones cuando sea posible.

Datos de Imágenes a Color

A la hora de usar funciones para conseguir los datos de las imágenes a color se pueden utilizar tres tipos de banderas:

- NUI_IMAGE_TYPE_COLOR: los datos obtenidos desde la cámara son en RGB

Ilustración 12. Imagen RGB

- NUI_IMAGE_TYPE_COLOR_YUV: se obtiene los datos de imágenes YUV pero convertidas a RGB32

Ilustración 13. Imagen YUV convertida a RGB

- NUI_IMAGE_TYPE_COLOR_RAW_YUV: se obtiene los datos de imágenes en YUV (YUYV) antes de su conversión a RGB32.

Ilustración 14. Imagen YUV

Estos tres tipos de imágenes son tomados por el mismo sensor de la cámara, comprimidos mediante un vector de bytes y enviados mediante el USB al ordenador. A continuación se descomprime y transforma al modo elegido.

Pese a que puede parecer que las dos primeras imágenes son idénticas (Imágenes 12 y 13 a la hora de tomarlas del vídeo se apreciaba en la imagen YUV convertida a RGB (imagen 13) un parpadeo aleatorio en ciertos píxeles.

Al ampliar ambas imágenes podemos notar una gran diferencia entre ambas imágenes (Figura 15) pese a que ambas están tomadas a la misma resolución (640x480). La máxima resolución (1280x960) solo está disponible para la opción TYPE_COLOR. La figura siguiente muestra la comparación entre las figuras 12 y 13 pero ampliadas para ver las diferencias.

Ilustración 15. Comparación entre RGB y YUV

Las dos primeras formas los formatos permanecían iguales (3 canales y 8 bits de profundidad), sin embargo, para poder tomar el ultimo tipo de imagen desde la cámara a color (RAW_YUV) había que convertir el formato a 1 canal y 16 bits de profundidad.

La forma de codificación de las imágenes es la siguiente. Cada imagen se compone de un conjunto de píxeles (más píxeles cuanto mayor sea la resolución). Cada píxel está compuesto por cuatro componentes: tres corresponden a los colores rojo, verde, azul. El cuarto componente corresponde a un valor de trasparencia (alfa) en el caso de imágenes RGBa y a un valor vacío en el caso RGB.

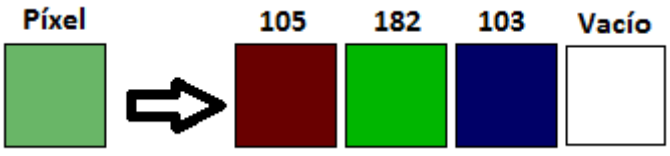

Ilustración 16. Descomposición de un píxel en RGB

Cada componente del píxel tiene un valor comprendido desde 0 hasta 255 correspondiente a un byte, donde los valores bajos corresponden a tonos oscuros; por tanto al tener cuatro componentes por píxel, el píxel en total estará compuesto por 4 píxeles.

La forma de organizar el vector de bytes que obtenemos del sensor es empezando por la esquina superior izquierda y terminando por la inferior derecha, avanzando de izquierda a derecha y de arriba abajo como se indica en la siguiente figura, donde los cuatro primeros bytes corresponden a los datos del primer píxel.

Ilustración 17. Vector de bytes RGB

Si la imagen en vez de ser en RGB es en RGBa entonces en el vector de bytes, el byte en blanco contendría el valor del factor de transparencia.

Datos de Profundidad

Existen dos modos de conseguir las imágenes de profundad. Estas son:

- NUI_IMAGE_TYPE_DEPTH: obtiene imágenes con los datos de profundidad.

Ilustración 18. Imagen de profundidad.

- NUI_IMAGE_TYPE_DEPTH_AND_PLAYER_INDEX: obtiene los datos con los datos de profundidad y además una capa con los píxeles correspondientes a cada persona detectada.

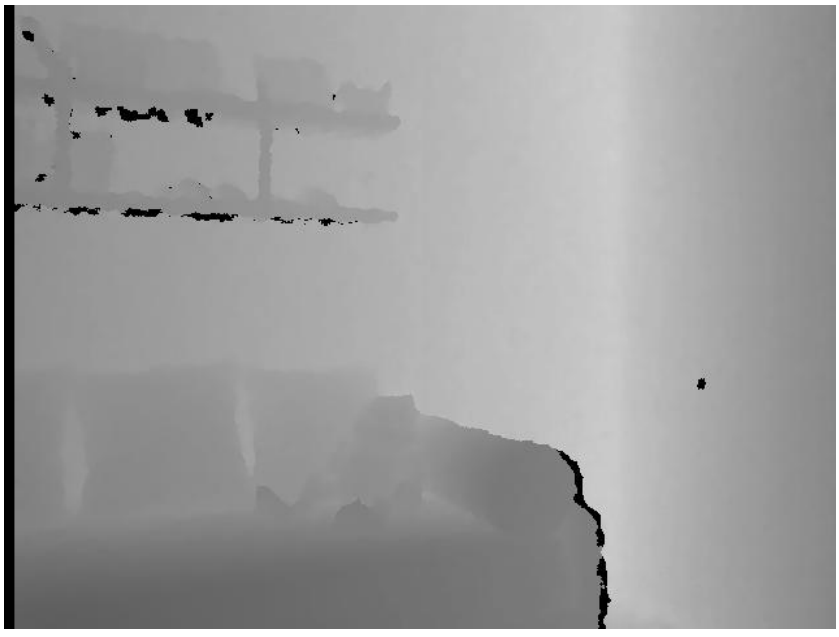

Ilustración 19. Imagen de profundidad con capa para personas

La diferencia principal radica en que si se inicia y utiliza la opción de TYPE_DEPHT obtenemos únicamente los datos de profundidad, mientras que si se utiliza la otra opción además obtendremos 3 bits que corresponderán al índice del jugador que aparece en la imagen, por lo que TYPE_DEPHT suele ser la opción elegida si únicamente queremos obtener los datos de profundidad.

A la hora de obtener los datos de profundidad de la cámara, éstos van también encapsulados en un vector de bytes. A diferencia del caso RGB, esos bytes no corresponden con las componentes de color de cada píxel, sino la distancia de ese punto a la cámara. Cada píxel se corresponde con dos bytes en el vector y su organización es de la misma forma que con la cámara RGB.

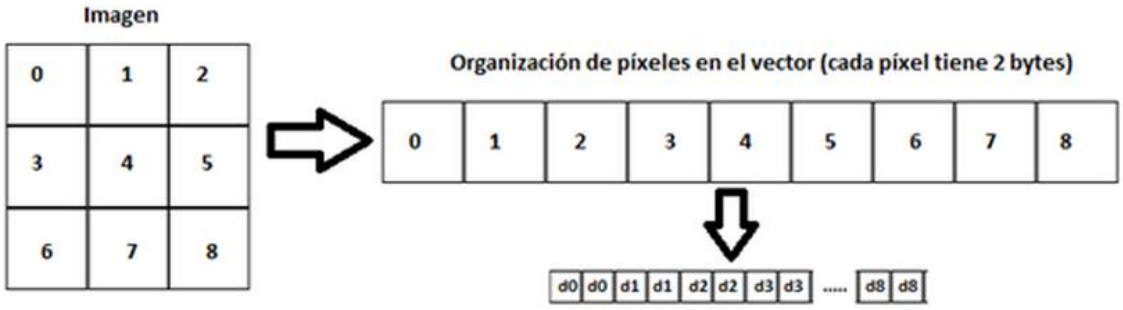

Ilustración 20. Vector de bytes profundidad

A la hora de calcular la distancia de un píxel en la imagen se deberá convertir los datos teniendo en cuenta la inicialización:

Si la opción elegida fue Depth entonces se hará una operación lógica OR con los bytes correspondientes al píxel, realizando antes un desplazamiento de 8 bits en el segundo byte del píxel.

Distancia (0,0) = (int)(Bits[0] | Bits[1] << 8);

Si se quiere calcular la distancia de los píxeles de una imagen de tipo DepthAndPlayer hay que tener en cuenta que los tres primeros píxeles corresponden al index del jugador.

Distancia (0,0) =(int)(Bits[0] >> 3 | Bits[1] << 5);

Datos de Segmentación de Personas

Tras haber obtenido la imagen de profundidad, dicha imagen se pasa por una serie de filtros que mediante unos clasificadores diferencia que pixeles de una imagen corresponden a una persona y cuales no. Kinect SDK es capaz de detectar hasta seis personas indexándolas.

Para desarrollar esos clasificadores utilizan características como vectores de desplazamiento y la diferencia de profundidad entre varios píxeles. Además sigue unas directrices como altura y anchura que debería tener.

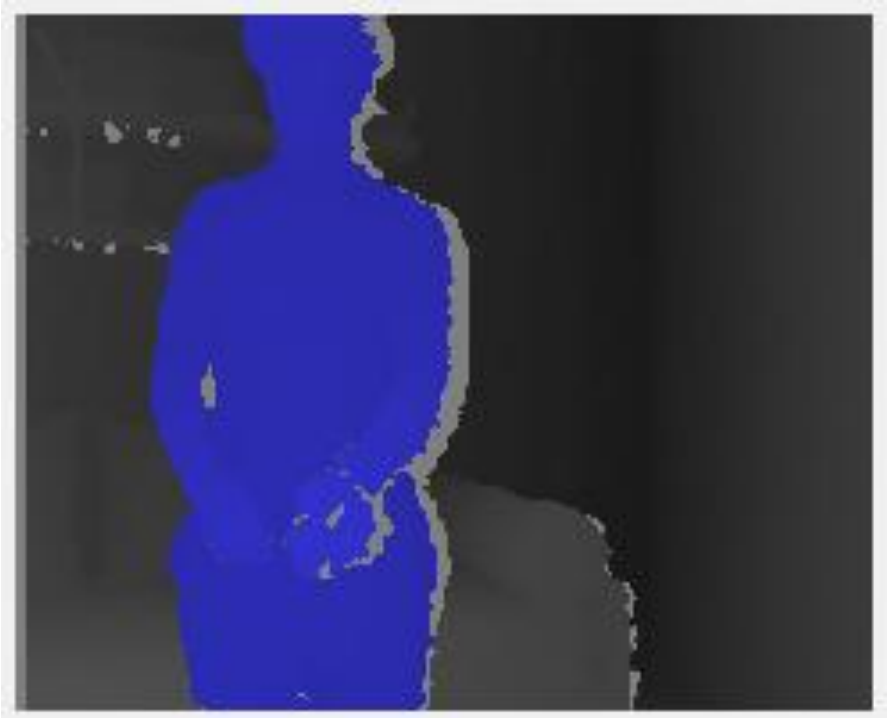

Ilustración 21. Capa jugador

Tracking del esqueleto

Kinect SDK es capaz de detectar la posición de hasta seis personas, pero solo es capaz de realizar el esqueleto de dos de ellas (las dos más cercanas). Aquellas personas a las que puede realizar el esqueleto las denominan activas, por lo que tendremos dos personas activas y cuatro pasivas.

Mientras que en las personas pasivas el único punto que se puede detectar es la posición (el centro de masas), además de su identificador, en las personas activas se puede obtener la estructura que contiene los 20 puntos detectados (véase figura 15).

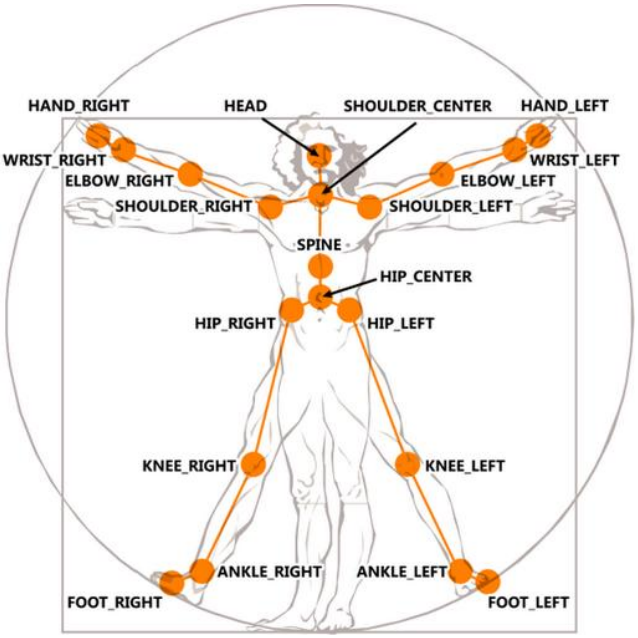

Ilustración 22. Posiciones del esqueleto

Para poder identificar posturas y posiciones parcialmente ocultas, Kinect cuenta 200 posturas comunes para poder rellenar los espacios ocultos.

La pregunta más importante es, ¿en qué se basa el algoritmo de Microsoft para realizar la esqueletización? Para ello emplearon una aproximación que consistía en reconocer cada parte del cuerpo mediante clasificación píxel a píxel en tiempo real. Para ello implementaron un algoritmo que procesa las imágenes de profundidad (con la ventaja de que no dependen de la intensidad de luz) y como no necesita información temporal para el funcionamiento del clasificador las analizará imagen a imagen.

Los datos para entrenar el clasificador fueron imágenes de profundidad con etiquetas en cada parte del cuerpo humano. Debido a la infinidad de poses, el proceso lo automatizaron utilizando datos de captura de movimiento de la Universidad de Carnegie Mellon para generar nuevas imágenes de entrenamiento además de algunas obtenidas desde la propia cámara.

Además para un aprendizaje mayor, por cada pose obtenida tomaban 12 imágenes a distintos ángulos.

4.2. OpenNI

PrimeSense es la empresa detrás de la tecnología de Kinect y lanzadora de un Framework de código abierto llamado OpenNI, que permite manejar el periférico de Microsoft. OpenNI (Interacción natural Abierta) es multiplataforma por lo que se pueden escribir aplicaciones tanto para Windows, Linux e IOS.

El termino de interacción natural muestra el concepto de interacción entre la persona y el dispositivo, donde para controlar el sistema lo haremos mediante control postural del cuerpo y la voz, sin necesidad de utilizar ningún otro periférico como ratón, mando o teclado. El control mediante OpenNI puede ser:

- *Speech and command recognition*, donde además de la cancelación de eco, podemos dar instrucción mediante comandos de voz.

- *Body Motion Tracking*, donde a partir de las imágenes de profundidad se detecta y analiza los cuerpos encontrados para realizar el esqueleto completo o parcial del usuario.
- *Hand gestures*, donde podremos utilizar las posiciones de la mano en la pantalla y respecto al cuerpo para controlar el dispositivo.

El propósito de OpenNI es crear una API estándar que haga posible la comunicación tanto con los sensores que nos proporcionarán las imágenes, como del array de micrófonos y el análisis de los datos.

Además de proporcionar datos, este framework permite grabar las secuencias de imágenes a color y 3D en un archivo con extensión propia (.oni). A la hora de crear este archivo se puede elegir diversos tipos de compresión.

Además de grabar, OpenNI permite la reproducción de estos archivos, por lo que si se tiene secuencias en las que aparecen personas, éstas serán detectadas y se podrá hacer simulaciones en las que por ejemplo, se realice el esqueleto del usuario como si en ese momento estuviera delante de la cámara.

A diferencia del software oficial de Microsoft que solo permite su utilización mediante Kinect, el software desarrollado por PrimeSense (OpenNI) permite su utilización mediante cualquier cámara que posea el chip PS1080 de PrimeSense.

Estas cámaras son:

- Microsoft Kinect
- ASUS Xtion PRO (sin cámara RGB)
- ASUS Xtion PRO Live (con RGB)
- PrimeSense PSDK

Todas estas cámaras contienen varias versiones, pero todas ellas tienen un gran parecido físico y componentes similares. A continuación se puede ver algunas de estas cámaras.

Ilustración 23. Cámaras con chip PS1080

4.2.1. Arquitectura.

OpenNI está compuesto por tres capas (véase figura 16):

- La capa superior (aplicaciones): representa la capa de desarrollo de aplicaciones.
- La capa intermedia (*interface*): es la conexión propia de OpenNI con los sensores y los middleware que analizan la información de los sensores y nos permiten obtener los datos para el uso en las aplicaciones.
- La capa inferior (*Hardware*): representa a los dispositivos de adquisición de imágenes (color, infrarrojos, y profundidad) y la captura del audio.

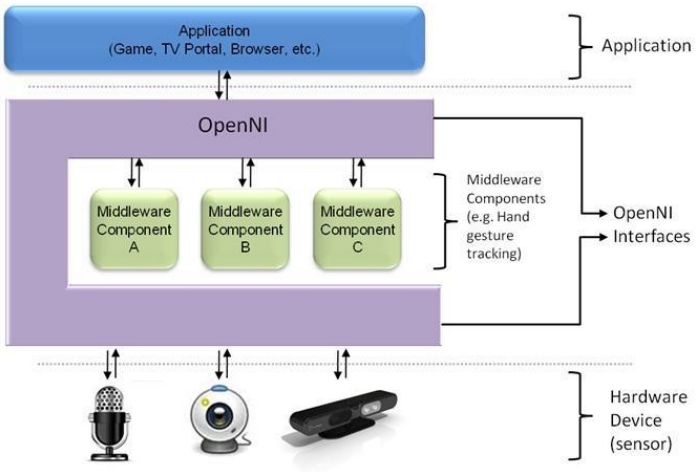

Ilustración 24. Arquitectura de OpenNI

La forma de trabajar con OpenNI es mediante la utilización de módulos, que son usados para producir y procesar los datos de los sensores. Estos módulos pueden ser tanto de dispositivos como de *middleware*.

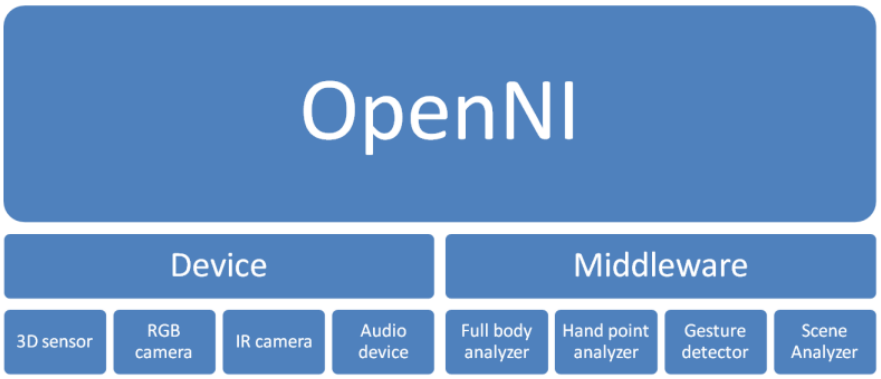

Ilustración 25. Módulos de OpenNI

Mediante el uso de este framework se tiene acceso a los datos de profundidad (3D sensor), como a las imágenes a color, infrarrojo y datos de audio. También permite acceso a los middleware desarrollados por PrimeSense como son:

- *Full body analyzer*: procesa los datos obtenidos por el sensor y genera información relacionada con el cuerpo como son las articulaciones, el centro de masas, orientación...
- *Hand point analyzer*: analiza los datos y genera la localización de las manos.
- *Gesture detector*: identifica gestos predefinido (por ejemplo, movimiento "clic" con la mano) y genera alertas a la aplicación.
- *Scene analyzer*: analiza las imágenes y proporciona información como las coordenadas del plano, identificación de la silueta de personas, y separación entre fondo y objeto en movimiento.

4.2.2. Tipos de imágenes.

Pese a que en la documentación oficial no aparece los formatos de tamaño y velocidad de adquisición de cada tipo de nodo generador de imágenes, OpenNI proporciona ciertas funciones para conseguir la información. Por lo que para conseguirla solo basta con incluir un pequeño código en el programa.

El código se deberá poner junto a la inicialización de los nodos y se podrá usar el siguiente código:

```
cout<<endl<<"Formatos permitidos por depth generator"<<endl;
XnUInt32 xNum2 = m_depthGen.GetSupportedMapOutputModesCount();
cout << "Support mode: " << xNum2 << endl;
XnMapOutputMode* aMode2 = new XnMapOutputMode[xNum2];
m_depthGen.GetSupportedMapOutputModes( aMode2, xNum2 );
for( unsigned int i = 0; i < xNum2; ++ i )
{
    cout << aMode2[i].nXRes << " * " << aMode2[i].nYRes << " @" << aMode2[i].nFPS << "FPS\n";
}
delete[] aMode2;
```

Donde m_depthGen es el nombre del nodo generador (en este caso de profundidad) que es elegido por el programador y que será remplazado para cada nodo.

Datos de Imágenes a Color

Se puede obtener imágenes de distintos tipos al igual que ocurría con las librerías oficiales. Cada tipo devuelve los datos de distinta forma y debe ser procesado de formas distintas para poder mostrar las imágenes. Todas las funciones para conseguir las imágenes son funciones miembros de los nodos generadores:

- GetMetaData()

Devuelve una imagen de tipo metadato en formato BGR por lo que para mostrarla por pantalla con unos colores adecuados deberemos cambiar los canales de orden. La imagen es de 8 bits por píxel y tres canales. A continuación se puede ver una imagen tomado en este modo.

Ilustración 26. Imagen de MetaDato

- GetRGB24ImageMap()

Este formato de imagen devuelve un puntero a una variable de tipo XnRGB24Pixel. Esta variable es propia de la librería OpenNI y en el fondo es una estructura de tres variables unsigned char que representan las tres componentes del color (RGB). La imagen obtenida está en formato BGR por lo que para mostrarlo por pantalla con los colores adecuados se deberá convertir a RGB.

Una de las desventajas de utilizar este tipo de imágenes es que de forma aleatoria y sin motivo directo la imagen se distorsiona y vuelve a su forma original.

- GetImageMap()

La función utilizada devuelve un puntero a una variable de tipo XnUInt8. Las imágenes obtenidas con esta función y las dos anteriores son idéntica una vez se han cargado y mostrado a través de las funciones de OpenCV.

- GetYUV422ImageMap()

Al igual que los casos anteriores, la imagen que se obtiene está en formato BGR. La imagen es idéntica pese a que el formato devuelto por la función es de tipo XnYUV422DoublePixel. Al igual que en RGB24, consiste en una estructura basada en variables de tipo unsigned char.

- GetGrayscale16ImageMap()

Los datos devueltos por esta función, al igual que los casos anteriores, corresponden a un puntero a una variable especial.

- GetGrayscale8ImageMap()

Los datos devueltos corresponden a un puntero que retorna una imagen idéntica a los casos anteriores.

El nodo generador de imágenes a color es capaz de proporcionar imágenes a distintas velocidades según sea el tamaño seleccionado:

- Imagen de 1280x1024 las velocidades son: 15fps
- Imagen de 640x480 las velocidades son: 30fps
- Imagen de 320x240 las velocidades son: 30fps, 60fps

Sin embargo, al utilizar las funciones mencionadas antes, el programa mostrará diez tipos de tamaños-velocidades pero algunos de ellos están repetidos.

Datos de Imágenes de Profundidad

Al igual que en el caso de las imágenes a color, para obtener los datos de profundidad se puede optar por conseguir directamente una imagen de metadato, o un puntero a la variable que contiene los datos de la imagen. Además se ha escalado la imagen teniendo en cuenta la profundidad máxima para obtener mejor resolución visual.

- GetMetaData
- GetDepthMap

La imagen que se obtiene es de 12 bits de profundidad en escalas de grises, donde los objetos más cercanos corresponden a colores más oscuros. Los otros cuatro bits corresponden a los datos sobre los usuarios.

Ambas formas de obtener las imágenes dan como resultado una imagen exactamente igual. La única diferencia es la forma de acceder a ella. Mediante la primera opción da como resultado una imagen de forma directa, mediante la segunda obtendremos un puntero por lo que usaremos esta forma cuando deseemos obtener imágenes en las que deseemos modificar la profundidad en función de la distancia. A continuación se puede ver las dos imágenes mencionadas antes.

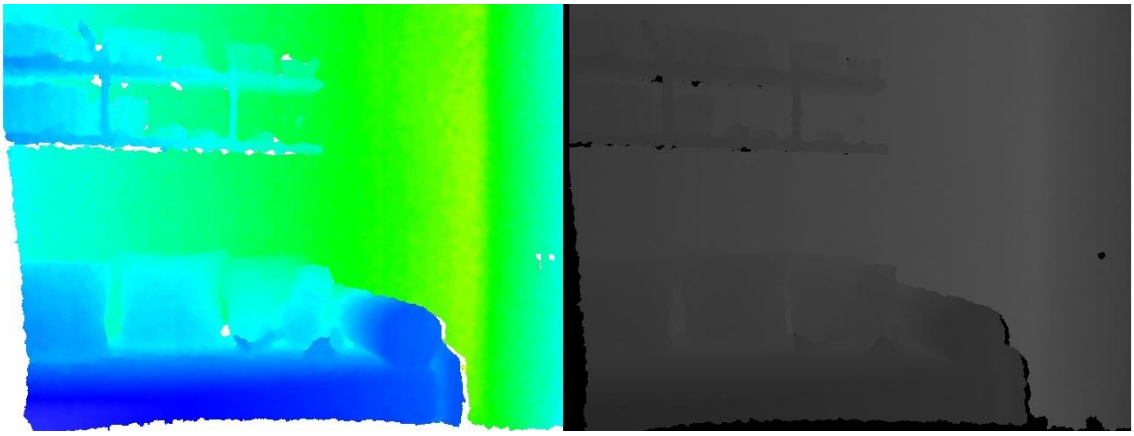

Ilustración 27. Imágenes de profundidad

El nodo generador de imágenes de profundidades es capaz de proporcionar imágenes a distintas velocidades según sea el tamaño seleccionado:

- Imagen de 640x480 las velocidades son: 30fps
- Imagen de 320x240 las velocidades son: 30fps, 60fps

Sin embargo, al utilizar las funciones para obtener tamaño y velocidades, el programa mostrará nueve tipos de tamaños-velocidades pero la gran mayoría repetidas al igual que en el caso de las imágenes a color.

Datos de Imágenes de Infrarrojo

Las imágenes de infrarrojo obtenidas son de 12 bits de profundidad. La cámara proyecta una serie de haces infrarrojos que son detectados y medida la deformación. En la imagen pueden ver la constelación de puntos emitidos y la imagen de infrarrojos resultante.

Ilustración 28. Imagen de infrarrojo

Como puede verse, la imagen está en escala de grises, sin embargo se diferencia claramente los objetos. La ventaja de este tipo de imagen respecto a la imagen de color reside en su no dependencia con la iluminación. Mientras que si la habitación tiene una iluminación deficiente la imagen a color no se verá, la imagen de infrarrojos no se verá afectado por ello, al igual que si hay una gran iluminación.

A la hora de usar las imágenes de infrarrojos hay que tener en cuenta un detalle, no pueden usarse al mismo tiempo la imagen a color e infrarrojo puesto que al usarlo a la vez la imagen se ralentizará y dará errores.

El nodo generador de imágenes de infrarrojos es capaz de proporcionar imágenes a distintas velocidades según sea el tamaño seleccionado:

- Imagen de 1280x1024 las velocidades son: 15fps
- Imagen de 640x480 las velocidades son: 30fps
- Imagen de 320x240 las velocidades son: 30fps, 60fps

Datos de Seguimiento de Personas

La segmentación y detección de personas es semejante a la utilizada por el driver oficial. Además, para obtener los píxeles que pertenecen a cada usuario será utilizada una función especial, GetUserPixels, la cual forma parte del nodo userGenerator.

Para poder utilizar la segmentación de personas se deberá haber inicializado los nodos tanto de usuarios como el de profundidad. No hay que olvidar que de los 16 bits de las imágenes de profundidad, solo 12 corresponden a la distancia y el resto identifica al usuario al que pertenece.

Si utilizamos estos datos podemos conseguir una imagen en la que la zona coloreada corresponde al usuario detectado.

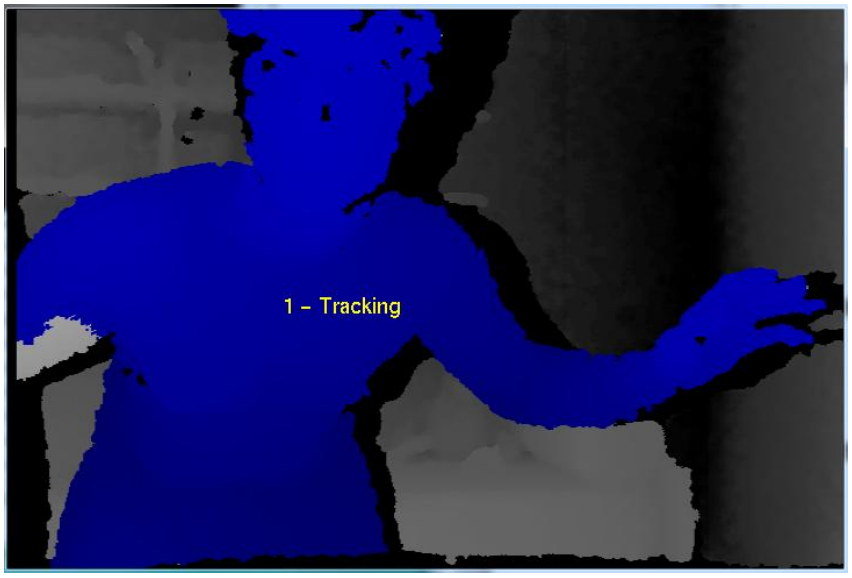

Ilustración 29. Imagen de profundidad con capa usuario

Tracking del Esqueleto.

A diferencia del software oficial, mediante el framework de OpenNI se puede realizar el esqueleto de más de dos usuarios al mismo tiempo, conociendo en todo momento qué esqueleto corresponde a cada usuario.

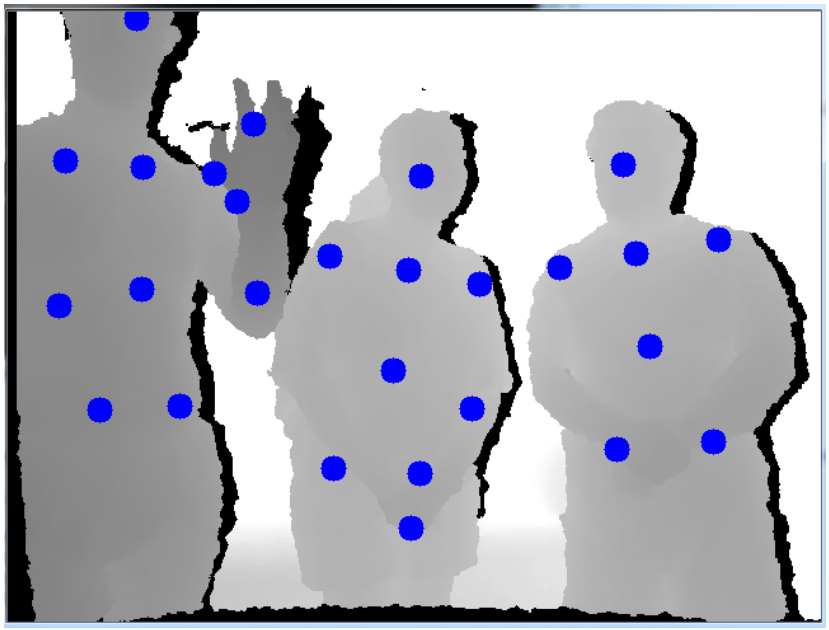

Ilustración 30. Imagen con los joint del esqueleto de los usuarios.

Mediante OpenNI se pueden detectar hasta 24 articulaciones por usuario. Además de la posición de cada articulación, también se obtienen datos sobre la rotación y la función de confianza de esa articulación, es decir, devuelve un valor entre 0 y 1 que corresponde con la probabilidad de que el punto detectado sea correcto, cuanto mayor sea ese punto, mayor probabilidad de acierto.

A la hora de saber la posición de una articulación no es necesario acceder a todo el esqueleto, sino que basta con acceder directamente a la información de una articulación en concreto. Las articulaciones y su posición en el array del esqueleto es la siguiente:

```
1  XN_SKEL_HEAD,
2  XN_SKEL_NECK,
3  XN_SKEL_TORSO,
4  XN_SKEL_WAIST,
5  XN_SKEL_LEFT_COLLAR,
6  XN_SKEL_LEFT_SHOULDER,
7  XN_SKEL_LEFT_ELBOW,
8  XN_SKEL_LEFT_WRIST,
9  XN_SKEL_LEFT_HAND,
10 XN_SKEL_LEFT_FINGERTIP,
11 XN_SKEL_RIGHT_COLLAR,
12 XN_SKEL_RIGHT_SHOULDER,
13 XN_SKEL_RIGHT_ELBOW,
14 XN_SKEL_RIGHT_WRIST,
15 XN_SKEL_RIGHT_HAND,
16 XN_SKEL_RIGHT_FINGERTIP,
17 XN_SKEL_LEFT_HIP,
18 XN_SKEL_LEFT_KNEE,
19 XN_SKEL_LEFT_ANKLE,
20 XN_SKEL_LEFT_FOOT,
21 XN_SKEL_RIGHT_HIP,
22 XN_SKEL_RIGHT_KNEE,
23 XN_SKEL_RIGHT_ANKLE,
24 XN_SKEL_RIGHT_FOOT
```

A pesar de que OpenNI está diseñado para la detección de 24 articulaciones, actualmente solo es capaz de reconocer 15 articulaciones, que son las que aparecen en la imagen siguiente:

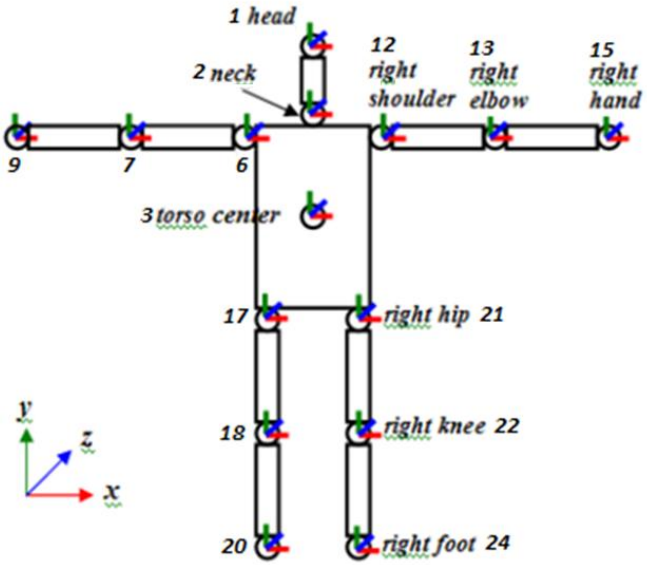

Ilustración 31. Posición de las articulaciones del esqueleto OpenNI

5. Esqueletización mediante cámara estéreo.

Durante el análisis de la librería OpenNI surgió una pregunta, ¿se podría utilizar los datos obtenidos por otro tipo de cámara, por ejemplo una estereoscópica, para obtener el esqueleto de las personas?

El origen de esta pregunta radica en la posibilidad de utilización de esta librería sin tener que tener la cámara para capturar los datos. Mediante OpenNI se puede grabar una secuencia de imágenes tanto de profundidad, como de imágenes a color. Una vez grabada la secuencia se puede utilizar para arrancar los programas desde ella obteniendo los mismos resultados que si tuviéramos la cámara conectada. A continuación se explicará el mecanismo y la conversión de las imágenes estéreo a formato OpenNI.

- Lo primero será la declaración de las variables que utilizaremos y la apertura del fichero que contiene los nombres de las imágenes estéreo.
- A continuación se inicializará la conexión con Kinect y se creará el nodo generador de imágenes de profundidad y el de grabación, seleccionando el nombre del fichero que se generará.
- Se creará un nodo ficticio de profundidad (MockDepthGenerator) que posibilitará la carga de datos por parte del usuario, además de cargar los datos de la conexión con la cámara; es decir, crea un nodo de profundidad al que se podrá pasar los datos elegidos por parte del usuario.
- Se procederá a la unión del nodo ficticio con el nodo de grabación seleccionando el tipo de compresión de los datos.
- Por último se obtendrá la imagen de disparidad y se cargará para grabarla en el lugar en el que se pondría los datos de la cámara Kinect.

Para utilizar el archivo grabado en cualquier programa que utilice los datos de profundidad solamente se necesita añadir dos funciones al código, una para abrir el fichero ".oni" y otra para buscar el nodo generador de imágenes de profundidad dentro del archivo.

Tras haber testeado el programa que lee secuencias de imágenes que previamente se ha grabado desde Kinect, se comprobará que la esqueletización se realiza sin problemas. Sin embargo, cuando se usa el archivo creado a partir de los datos de la cámara estereoscópica se observa que cuando un usuario aparece por pantalla no es detectado.

El motivo por el que el framework no es capaz de detectar al usuario es debido a que la función para conseguir el esqueleto no funciona como se pensó en un primer momento.

La función para conseguir el esqueleto necesita que se haya inicializado previamente la obtención de las imágenes de profundidad. Al obtener las imágenes de profundidad solo 12 de los bits corresponden a la distancia, y los otros 4 bits a si ese bit corresponde a una persona o no. Como OpenNI necesita de esos datos para generar el esqueleto y los datos se generan por la cámara, hace que sea imposible la utilización de la cámara estéreo.

6. Comparación entre SDK y OPENNI

En este apartado se realizará una comparación entre ambas librerías para la obtención de las imágenes y detección de personas.

Licencia de uso

La primera diferencia destacable para poder usar estas dos tecnologías es la licencia de uso. Mientras que para poder usar la Kinect mediante el SDK debe hacerse mediante unas condiciones propias impuestas por el equipo de Microsoft, para poder usarla mediante OpenNI se hará bajo licencia pública general de GNU. En ambos casos será posible usar los programas que desarrollemos con fines comerciales siempre que no usemos la versión Kinect SDK beta.

Método de instalación

A la hora de instalar una librería u otra se puede encontrar otra gran diferencia en cuanto a sencillez. Para instalar las librerías oficiales (SDK) basta con seleccionar la versión que se desee y descargar un ejecutable que nos guiará paso a paso en la instalación. Para la instalación del framework de PrimeSense es necesario seleccionar los archivos que se descargarán desde la página oficial.

A diferencia de la instalación del SDK, que es bastante clara y sencilla, la instalación de OpenNI es muy confusa y ambigua pues dependiendo del lugar que consultemos indican un orden de instalación u otro. Sin embargo en la mayoría de los sitios aparecen en este orden, descargando siempre las versiones inestables:

- OpenNI Binaries
- SensorKinect091-Bin-Win32
- OpenNI Compliant
- OpenNI Hardware

Además de la instalación de los archivos anteriores habrá que verificar que las siguientes variables de entorno del sistema contienen las siguientes direcciones.

Variable	Valor
CLASSPATH	C:\Program Files (x86)\OpenNI\Bin\org.OpenNI.jar
OPEN_NI_BIN	C:\Program Files (x86)\OpenNI\Bin
OPEN_NI_INCLUDE	C:\Program Files (x86)\OpenNI\Include
OPEN_NI_INSTALL_PATH	C:\Program Files (x86)\OpenNI\
OPEN_NI_LIB	C:\Program Files (x86)\OpenNI\Lib
PATH	C:\Program Files (x86)\OpenNI\Bin;C:\Program Files (x86)\PrimeSense\NITE\bin

A pesar de seguir esos pasos, en muchas ocasiones al conectar la cámara puede que no la reconozca o no consigamos que funcionen los programas. Pero gracias al paquete de instalación de ZigFu para OpenNI se puede realizar una instalación similar a la que se hace con el SDK, facilitando la instalación a costa de no utilizar siempre las últimas versiones existentes de los componentes anteriores.

Plataforma y entorno de desarrollo

Otra gran diferencia entre estas dos librerías es el tipo de plataforma sobre la que se puede trabajar y los lenguajes de programación admitidos. Mientras que para usar el SDK es obligatorio trabajar con Windows 7 nativo y usar la versión de Visual Studio 2010, para la utilización de OpenNI no es necesario ningún entorno de desarrollo determinado y puede utilizarse tanto en Windows y Linux; por lo que su amplitud de potenciales candidatos aumenta.

Lenguajes de programación

En cuanto a los lenguajes de programación, se pueden utilizar tanto C, C++, C#, VB y Java en caso de utilizar SDK y solamente C++ y C# si se utiliza OpenNI.

A pesar de poder utilizar dos tipos de lenguajes con OpenNI, en la documentación oficial que es accesible al instalar el framework, solo aparecen las funciones de C++.

Solamente se encontrará la información de C# dentro de algún ejemplo y en foros por internet, por lo que en su mayoría si se buscan ejemplos mediante la utilización de esta librería se encontrarán en el lenguaje C++.

A diferencia de lo que pasaba en el caso anterior, en la documentación oficial que obtenemos al instalar el SDK, se pueden ver las funciones para programar tanto en C++ como C# por lo que no dependemos tanto al tener más información. La tendencia en el caso de SDK es a utilizar el lenguaje C#, puesto que se reduce el código necesario para realizar la misma aplicación que si usáramos C++. Además la mayoría de ejemplos y aplicaciones que se pueden encontrar por internet están escritos en este leguaje. Otro de los motivos por los que los usuarios del SDK se decantan por la programación en C# es debido a la utilización de MFC para otorgar a las aplicaciones de un entorno gráfico más amigable, por lo que la alternativa de C# es más fácil y sencilla.

Tipos de imágenes

Los tipos de imágenes que se obtienen con una u otra librería son similares:

- Imágenes a color
- Imágenes de profundidad
- Obtención de la capa de usuario
- Obtención de las articulaciones de un usuario

A pesar de que tanto para SDK y OpenNI se pueden obtener las imágenes de color y profundidad de diversos tipos, solamente se puede observar diferencias entre las imágenes para el caso del SDK. Para OpenNI la única diferencia entre la obtención de un tipo u otro es el modo de retorno de la imagen, por lo que existe un bug por el cual la imagen devuelta siempre es del mismo tipo, aunque la función indique lo contrario.

Ilustración 32. Comparación imágenes de color OpenNI - SDK

Como puede verse en la imagen anterior (Ilustración 32), tanto la imagen obtenida desde OpenNI (izquierda) como la obtenida por SDK (derecha) son idénticas y eso es debido a que el proceso de obtención de las imágenes corresponde con una de las capas inferiores de Kinect que no se ven afectadas por el framework elegido. Otra muestra de esto es la obtención de las imágenes de profundidad (véase ilustración 33).

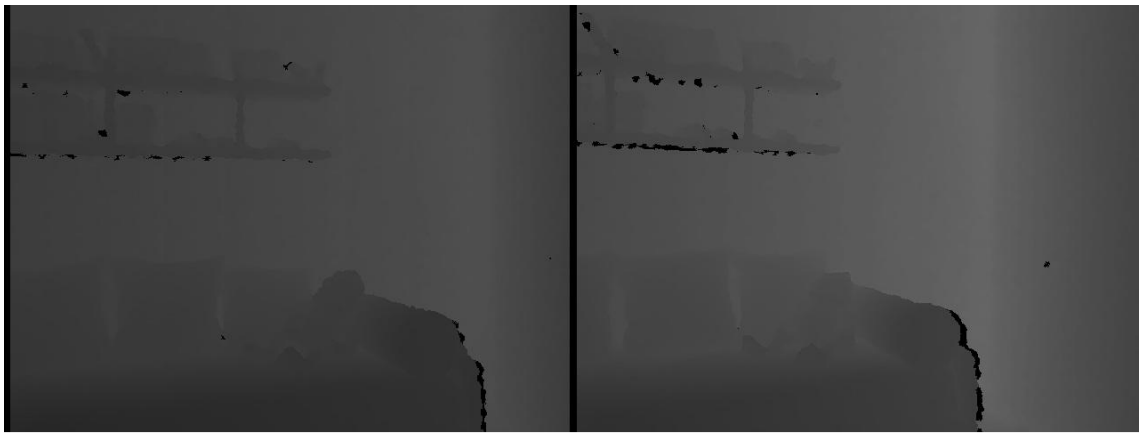

Ilustración 33. Comparación imágenes de profundidad OpenNI – SDK

Una de las ventajas del uso del framework de PrimeSense es la capacidad de obtención de las imágenes de infrarrojo, dotando a los programas de una "visión nocturna" debido a su no dependencia de la luz. Pese a que las imágenes de profundad son invariantes a la iluminación, no proporcionan tanta información visual como las imágenes de infrarrojos, que se encuentran a medio camino entre el color y la

profundidad, pues podemos encontrar cierta diferencia entre aquellos objetos con colores distintos.

Ilustración 34. Imagen infrarrojo de OpenNI

Fiabilidad en la obtención y esqueletización del usuario

Tras obtener las imágenes de profundidad con la capa de usuario en ambas tecnologías se observa que, pese a que la imagen de profundidad es la misma, la capa de usuario es distinta. Para comprobar las diferencias entre SDK y OpenNI se han tomado varias imágenes en distintas situaciones: interior (con y sin luz) y exterior.

Obtención en interior a oscuras:

Al obtener las imágenes se pudo observar la gran ventaja de la cámara frente a las convencionales. La detección y esqueletización del usuario se realiza de forma correcta en ambos casos.

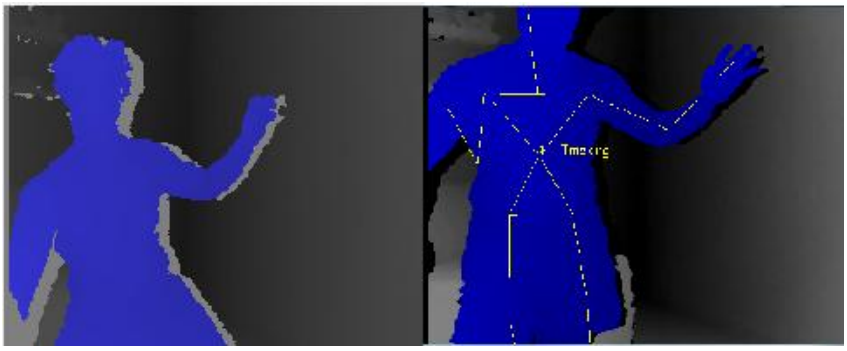

Ilustración 35. Comparación Interior a Oscuras

Como puede verse en la imagen anterior (Ilustración 35), la capacidad para la detección de la silueta de OpenNI (derecha) es mejor. A la hora de seguir las el tracking del usuario, ambos sistemas eran bastantes buenos. Sin embargo el driver oficial de Kinect era significativamente mejor analizando las partes visibles del usuario. La ventaja del driver no oficial reside en la capacidad de su función de confianza de los joint analizados, dando un porcentaje de fiabilidad de ese punto, por lo que se puede mostrar un esqueleto parcial con la certeza de que los puntos utilizados son buenos.

Obtención en interior con iluminación alta:

Se realizó esta otra prueba para comprobar si la luz en el interior de una habitación puede modificar la capacidad de detección y observar si para la realización de una aplicación habría que tener esto en cuenta. Además se encendieron algunas lámparas para incrementar la luz y así comprobar como afecta un exceso de iluminación.

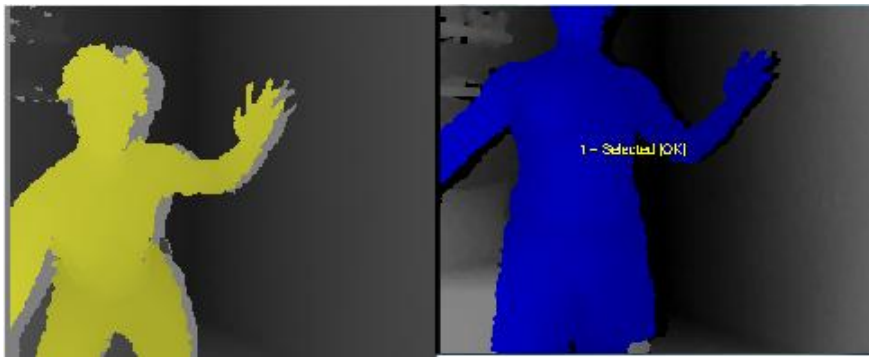

Ilustración 36. Comparación Interior con Iluminación.

Si se comparan las imágenes tomadas en el interior, tanto a oscuridad total, como con gran iluminación, los resultados no varían.

A la hora de realizar el tracking del usuario ocurría exactamente lo mismo que en el caso anterior, la esqueletización se hacía de forma correcta en ambos casos siendo más precisa en el caso de los driver oficiales.

Obtención en exterior:

Tras el éxito de la cámara para la detección de personas en el interior, se procedió a realizar pruebas en el exterior para comprobar el grado de eficacia de la cámara.

Para ello se puso la cámara en el exterior y una persona a distintas distancias de la cámara. Tanto la cámara como la persona recibían de forma directa los rayos solares.

Se comprobó que en este caso el sensor de profundidad no era capaz de obtener ninguna imagen, por lo que ninguna de las dos librerías eran capaces de detectar al usuario.

Después del fracaso obtenido en el exterior, se intentó realizar alguna otra prueba en la que los resultados mejorasen. Para ello se situó la cámara de forma que no incidieran sobre ella ningún rayo solar y al usuario a distintas distancias en una zona soleada.

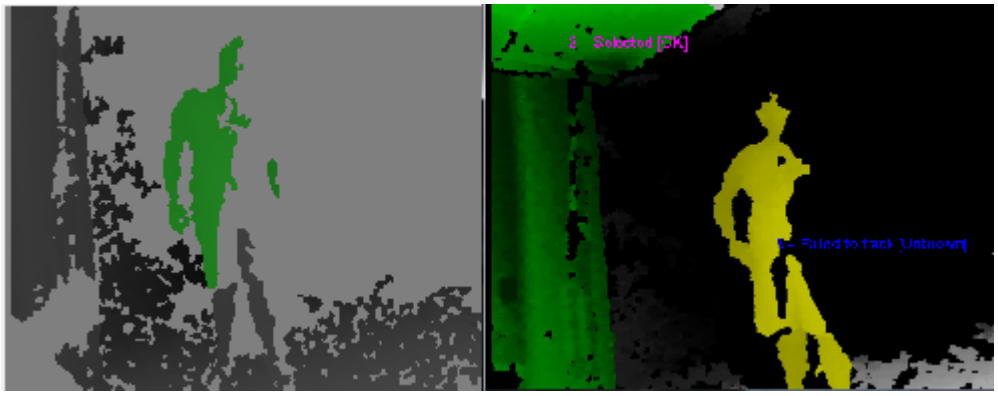

Ilustración 37. Comparación Exterior.

Como puede verse los resultados obtenidos no son demasiados buenos. La distancia a la que se situaba el usuario eran de apenas 2 metros y la rapidez con la que ambos sistemas dejaban de detectar era muy alto.

En la imagen correspondiente a OpenNI (derecha) puede verse mejores resultados en la capa del usuario. Sin embargo el sistema también detectaba numerosas superficies (pared y techo) de forma errónea y esto se debe a que las restricciones que utiliza para la detección de que es una persona y que no son mucho más flexibles, lo que induce en algunos casos a error.

En cuanto a la esqueletización el sistema de Microsoft realizaba el esqueleto a pesar de cometer bastantes errores debido a la deficiente información. Sin embargo, el sistema de PrimeSense no era capaz de realizar el tracking del usuario.

Otras diferencias

Una de las grandes ventajas que posee el SDK de Microsoft es la capacidad de acceso al motor que viene incorporado con la Kinect, pudiendo mover verticalmente el dispositivo según nuestro gusto.

La gran ventaja de poder acceder al motor radica en la posibilidad de poder encuadrar al usuario detectado para poder ver lo máximo posible del usuario, o realizar un barrido cuando sea requerido.

Por otro lado OpenNI tiene la ventaja de poder implementar y testear el programa sin necesidad de tener conectada la cámara. La importancia de esto se encuentra en la posibilidad de realizar programas sin tener que adquirir una cámara o sin la tediosa necesidad de estarse levantando cada vez que se quiera probar una nueva versión del programa.

7. Usos y aplicaciones de Kinect

Durante este capítulo se pueden ver las numerosas aplicaciones para las que se está utilizando la cámara, que va desde la diversión hasta aplicaciones en medicina.

Videojuegos para XBOX 360

En su origen el diseño y fabricación de está cámara estaba orientada en exclusiva al control de los videojuegos desarrollados por Microsoft para su videoconsola XBOX.

Se han desarrollado unos 80 videojuegos en los que el control se realiza mediante reconocimiento de voz y posturas corporales, haciendo que el usuario se involucre más en el juego y participe de manera más activa.

Existen títulos desarrollados tanto para los más pequeños (a partir de 3 años), como para los mayores de edad (a partir de 18), centrando sus esfuerzos en la franja comprendida desde los 3 a los 12 años (87% de los títulos).

Además de su variedad en edades también se pueden encontrar gran variedad de géneros en los que el punto principal es la actividad física del individuo, ya sea luchando, bailando, o controlando y moviendo a algún personaje. En el siguiente link se puede ver un video demostrativo la gama de Xbox.

http://www.youtube.com/watch?v=_xDZNSuOQiw

Asistente para aparcamiento de vehículos

Navegando por internet se pueden encontrar aplicaciones desarrolladas por algunos usuarios en los que, apoyándose en tanto en la cámara de RGB como la de profundidad, conseguían que un programa les avisara cuando tenían un objeto cerca del coche, de forma parecida a la que hacen los sistemas actuales de ultrasonidos.

La ventaja al usar esta cámara está en la posibilidad de ver visualmente la parte trasera del vehículo pudiendo ver la parte baja del vehículo que no se ve por el espejo retrovisor interno.

Además, mediante la cámara de profundidad se puede hacer una reconstrucción en tiempo real de los obstáculos, identificando la distancia y tamaño a la que se encuentran. En el link siguiente pueden ver un vídeo demostrativo:

http://www.youtube.com/watch?feature=player_embedded&v=Y3M61Qor8-g

El desarrollador de esta aplicación fue Gibson Hu, estudiante de postgrado de la Universidad de Tecnología de Sydney (Australia). Para el desarrollo de esta aplicación se basó en el software libre: Ubuntu, OpenCV y OpenKinect (otra de las librerías de Kinect).

Desarrollo de prótesis ortopédicas

Aunque parezca sorprendente, Kinect también se ha llevado al campo de la medicina y presentado ante un concurso promovido por Siemens en 2011.

En dicho concurso, dos jóvenes estudiantes de la Universidad de George Washington mostraron su investigación en bioingeniería para el desarrollo de aplicaciones que ayuden a entender los patrones al caminar de gente discapacitadas que utilicen prótesis ortopédicas.

Mediante el análisis del movimiento de dichos individuos se pretende conseguir un tratamiento que mejore la sanación de sus heridas, así como la posibilidad de rehabilitación desde su casa, ahorrando costes médicos.

Para más información se puede consultar el siguiente link:
http://www.levelup.com/noticias/16544/Kinect-podria-ayudar-a-desarrollar-protesis-ortopedicas/

Visualización en quirófanos

Un equipo del hospital "Toronto's Sunnybrook Hospital" ha desarrollado una aplicación que les permite consultar el historial e imágenes de los pacientes sin tener que salir de la sala de operación. Para la manipulación de las imágenes del paciente utilizan su propio cuerpo para avanzar, retroceder o hacer zoom, permitiendo la consulta por parte del cirujano.

La ventaja de utilizar este método es la reducción del tiempo de operación: se reduciría los componentes que necesitan esterilización, siendo imposible meter un ordenador dentro de un quirófano; la pérdida de tiempo por parte del cirujano sería considerable, debido al tiempo que tarda desde que sale, se esteriliza y vuelve a entrar.

Existe una empresa que ha decidido implementar y comercializar esta idea, se trata de Tedesys, una empresa cántabra que ha bautizado a la aplicación con el nombre de TedCas. A continuación se puede ver un ejemplo de Kinect usada en los quirófanos: http://www.youtube.com/watch?v=Lt5Dv250rPM&feature=player_embedded

Tratamiento parálisis cerebral

La medicina y los videojuegos en ocasiones pueden ir de la mano. Un grupo de la Universidad CEU Cardenal Herrera de Valencia ha desarrollado un programa para aumentar la movilidad de personas con parálisis cerebral mediante el ejercicio.

Dicho programa captura los movimientos de los pacientes trasladándolos al juego. Según la patología del paciente, se seleccionará que parte del cuerpo debe trabajar asignando las velocidades y grado de dificultad del juego.

Cada sesión queda registrada en la ficha del paciente, que controla su progresión de velocidad, distancia… para evaluar la mejora de sus capacidades de forma desglosada. Se pude ver más información en el siguiente link:

http://www.agenciasinc.es/Noticias/Investigadores-de-la-UCH-aplican-el-sensor-Kinect-en-tratamientos-de-paralisis-cerebral

Control del peso

Un equipo de investigadores de Eurecom, escuela de ingeniería e investigación situada en Sophia Antipolis (Francia) ha desarrollado una aplicación que mediante su algoritmo de análisis de varios parámetros del cuerpo humano son capaces de evaluar el peso de una persona mediante la utilización de la cámara Kinect.

El sistema funciona de la siguiente manera: cuando la persona se sitúa a la distancia óptima de la cámara (unos 2 metros), ésta captura varios puntos de referencia para determinar el género del individuo, altura, longitud y circunferencia de los brazos, cintura, piernas y cuello. Las medidas obtenidas se comparan con una base de datos antropométricos del National Health and Nutrition Examination Survey, cuya base de datos cuentan con medidas de unas 28000 personas.

Esta aplicación a captado el interés del Center for Human Space Robotics de Turín, los cuales desean aprovechar la capacidad de esta aplicación para sustituir su actual sistema de medida del peso de los astronautas en gravedad cero. El sistema actual consiste en una silla con un muelle oscilante, por lo que el sistema es mucho más voluminoso y pesado que si usaran la aplicación con Kinect. Ambos sistemas tienen un error aproximado de unos 3 Kg.

Sin embargo, no solo el Eurecom se ha dado cuenta del potencial de Kinect para calcular el volumen y forma corporal. La empresa WLAM también ha desarrollado una aplicación para escanear el cuerpo.

WLAM pretende utilizar Kinect para el análisis corporal y poder conseguir registrar los cambios y evolución en el cuerpo de un deportista o una persona que realice una dieta.

La aplicación de momento se puede usar de forma gratuita debido a que de momento siguen en fase de pruebas. Para acceder se puede entrar en el siguiente link: https://enforme.vlam.ca/Welcome/login

Generación de muñecos personales

Tras la gran expansión de Kinect y el auge de las impresoras en 3D, capaces de imprimir figuras en plástico, no era de extrañar que ambas cosas terminaran fusionándose.

La idea consiste en utilizar los datos de profundidad de varias cámaras para conseguir obtener la figura tridimensional deseada. A continuación se utiliza un programa de diseño tridimensional y se exporta a algún formato admitido por las impresoras, que en cuestión de minutos reproducen la figura.

La capacidad de este sistema se mostro tanto en Londres como en Barcelona durante el año 2011 para mostrar las posibilidades de las nuevas tecnologías. Además los viandantes podían llevarse su propia figura de recuerdo. A continuación se puede ver un video demostrativo:
http://www.youtube.com/watch?feature=player_embedded&v=WeJoVSPFdwQ

Compra en escaparates virtuales

Aparte de su aplicación TedCas para la visualización en quirófanos, la empresa Tedesys posee otro producto llamado TedShop. Este producto consiste en un escaparate virtual que se manejará mediante Kinect y el movimiento del usuario.

En dicha aplicación el cliente puede elegir entre los productos de la empresa y elegir modelo y talla para saber si tienen lo que desea. La aplicación cuenta con la capacidad de rápida actualización por parte del comerciante. La aplicación ya está en uso en algunos establecimientos de España.
http://www.youtube.com/watch?v=ca0qWl2Hf2g

También se están desarrollando otros tipos de tiendas virtuales, como la ideada por el ESNE, en el que el usuario podrá seleccionar un objeto y ver en pantalla como quedaría en el usuario.

Estudios geográficos

Algunos de los usos de Kinect están destinados a la investigación. Ken Mankoff, doctorado de la Universidad de California, ha desarrollado un programa capaz de hacer escaneos rápidos en 3D de cavernas heladas dentro de glaciares. Durante sus trabajos lleva siempre consigo su Kit (véase ilustración 30).

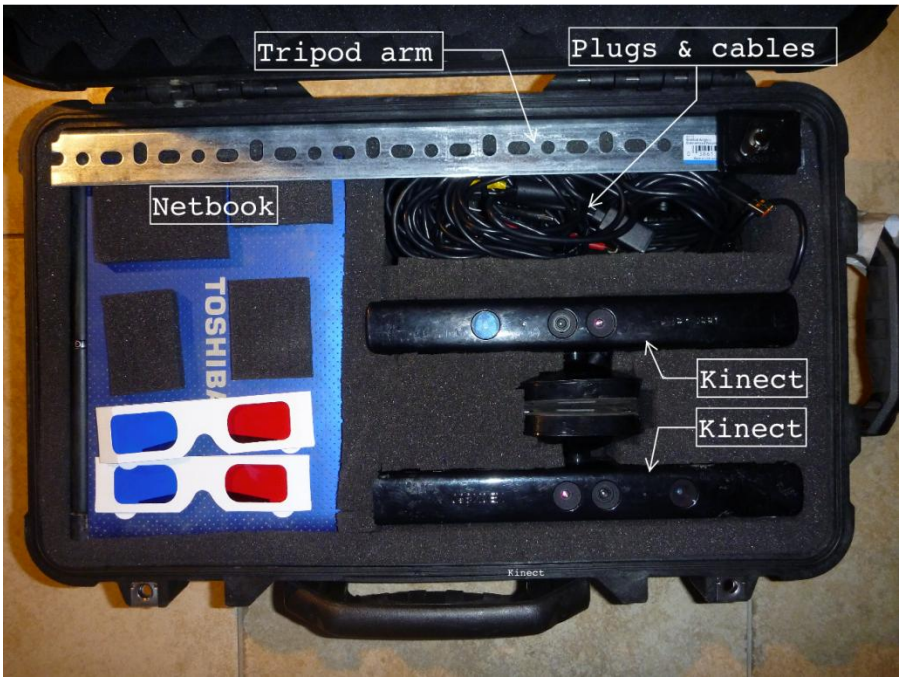

Ilustración 38. Kit de investigación

Durante su jornada de trabajo graba una gran cantidad de datos, que tras la grabación analizará para poder conseguir un mayor grado de precisión

Marco Tedesco de la Universidad de Nueva York pretende usar el dispositivo junto a un pequeño helicóptero o barco a control remoto para realizar la cartografía de lagos de agua en deshielo. Para más información se puede visitar la página del autor: http://kenmankoff.com/tag/kinect

Control de robot mediante comando corporales

El rango de aplicaciones para la cámara no tiene fin. Algunos desarrolladores la han utilizado para teleoperar robots de formas diversas.

Algunos desarrolladores han optado por manejar el robot desde una plataforma móvil (cinta para correr). Mediante el avance o retroceso desde el punto medio de la cinta el usuario controla el avance del robot. Mientras que la cámara detecte que el usuario se encuentra avanzando en la parte delantera de la cinta el robot avanzara. Así mismo, si detecta que el usuario gira sobre si mismo (determinando que hombro se encuentra más cerca y cual más lejos) el robot comenzará a girar hacia el mismo lugar que el operador. Puede verse un video de esta implementación en el enlace siguiente: http://www.youtube.com/watch?v=DEziMqYbpEs&feature=player_embedded

Otra alternativa a la teleoperación es el reconocimiento de poses para modificar las rutinas del robot. De tal forma que si levantamos un brazo podemos parar la acción actual, si levantamos una pierna indicamos cambio de rutina… De esta forma con unos cuantos comandos podemos hacer una gran cantidad de rutinas distintas y que no se solapen unas con otras. La ventaja de usar este método frente al anterior es la no dependencia del uso de una plataforma, además del aumento de precisión en los movimientos. En el siguiente link hay un ejemplo de este tipo de teleoperación: http://www.youtube.com/watch?v=GdepIXZTJsw&feature=player_embedded#!

La ventaja de poder teleoperar robot mediante movimientos posturales en vez del uso de un mando radica en la sencillez de pasar de un tipo de movimiento a otro, por lo que podría usarse para mover robots en ambientes peligrosos como por ejemplo el manejo de brazos robóticos para manipular objetos peligrosos para el ser humano, ya sea por ser productos tóxicos, por su peso elevado o materiales explosivos.

8. Conclusiones

Tras haber estudiado dos de las posibilidades para la obtención de los datos de la cámara Kinect y analizado las funciones se ha realizado una valoración sobre los puntos a favor y en contra de ambas tecnologías.

Uno de los puntos a favor del uso de los driver de PrimeSense es su carácter multiplataforma, por lo que si se desea programar en un sistema operativo distinto de Windows 7 se deberá utilizar este driver.

Si por el contrario se utilizará Windows 7 y no es necesario realizar el tracking de más de 2 personas al mismo tiempo, la mejor opción es el uso del SDK de Microsoft, debido a su fiabilidad de obtención del usuario y su rapidez a la hora de captarlo.

Otro punto a favor del uso del SDK es la última versión lanzada por Microsoft que permite realizar el reconocimiento facial de los usuarios. Además posee la opción de la utilización de la función Near y la incorporación del tracking de usuarios sentados, por lo que solamente tomará los datos de la parte superior del usuario.

La facilidad de uso del SDK es sorprendente además de su numerosa documentación, lo que facilita su aprendizaje (sobre todo a programadores de C#) y su gran implantación por lo que muchas empresas se lanzan a su uso.

Ya se ha visto la gran capacidad de la cámara en el interior y el gran número de aplicaciones desarrolladas. El futuro de la cámara es prometedor pese a que todavía estamos al comienzo de expansión de esta tecnología.

9. Posibles líneas de futuros trabajos

En este apartado se explicar algunas propuestas para proyectos futuros en los que poder utilizar las funcionalidades de esqueletización de la cámara Kinect:

- Para aprovechar una de las mayores ventajas de la cámara, la esqueletización, se podría realizar un proyecto en el que mediante posiciones corporales controlar los movimientos realizados por el robot.

- Otra de las posibilidades aprovechando la esqueletización del usuario podría consistir en el desarrollo de una aplicación para el reconocimiento de diversas actividades como podría ser la detección de personas sentadas o caminando de forma normal. El objetivo de distinguir actividades podría ser la detección de situaciones anómalas como caídas de personas.

- Otra alternativa sería la utilización de Kinect para la realización de una aplicación de realidad aumentada, donde el usuario pueda moverse entre las distintas páginas de la universidad y ver toda la información contenida en ella.

Presupuesto

El presupuesto para la realización del proyecto puede variar según si se elige la versión gratuita de Visual Studio o la versión completa y de pago.

Para la realización del proyecto se utilizó la versión gratuita por lo que el precio de realización del proyecto baja considerablemente.

Tanto el paquete de Microsoft, PrimeSente y OpenCV son gratuitos por los que no se tendrán en consideración para calcular el presupuesto.

Además se considera que el precio del ordenador no se tendrá en cuenta.
Los precios que se han tenido en cuenta son los siguientes:

- Cámara Kinect con cable de conexión al ordenador

  Precio de la cámara: 150 €

- Horas de inversión para la realización del proyecto

  Tiempo empleado: 430 horas

  Precio hora de ingeniero: 30€/hora

  Precio por realización: 12 900 €

Por tanto el precio del presupuesto final es de **13 050 €**

Bibliografía

[1] Página oficial de Visual estudio: http://msdn.microsoft.com

[2] Bradski, G. and Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[3] Blog oficial de Kinect: http://blog.kinectfordevelopers.com
http://www.microsoft.com/en-us/kinectforwindows

[4] Kinect for Windows Software Development Kit

[5] OpenNI User Guide

[6] González Jiménez, J. *Visión por Computador*. Paraninfo, 2000.

[7] Shotton, J. Fitzgibbon, A. Cook, M. Sharp, T. Finocchio, M. Moore, R. Kipman, A. Blake, A. *Real-Time Human Pose Recognition in Parts from Single Depth Images.* Microsoft Research Cambridge & Xbox Incubation

[8] Jacob, A. *Kinect weighs astronauts just by looking at them*. New Scientist magazine, 24 Diciembre 2011.

[9] Mann, A. Scientists Hack Kinect to Study Glaciers and Asteroids. Wired Sciencie, 14 Diciembre 2011.

[10] Plagemann, C. Ganapathi, V. Koller, D. Thrun, S. *Real-time Identification and Localization of Body Parts from Depth Images*. Artificial Intelligence Laboratory, Stanford University.

Nota: todas las páginas web se encontraban disponibles a fecha 23 de agosto de 2012.